PortSIP IVR SDK User Manual for Visual Studio .NET (P/Invoke)

Compatibles with Windows 2000/XP/2003/Vista/7/2008.

http://www.portsip.com

V8.1, Dec 26, 2012

portsip
VoIP Solutions Company

NOTICE

© 2012 PortSIP Solutions, Inc. All intellectual property rights in this publication are owned by PortSIP Solutions, Inc. and are protected by United States copyright laws, other applicable copyright laws, and international treaty provisions. PortSIP Solutions, Inc. retains all rights not expressly granted.

This publication is PortSIP confidential. No part of this publication may be reproduced in any form whatsoever or used to make any derivative work without prior written approval by PortSIP Solutions, Inc.

PortSIP Solutions, Inc. reserves the right to revise this publication and make changes without obligation to notify any person of such revisions or changes. PortSIP Solutions, Inc. may make improvements or changes in the product(s) and/or the program(s) described in this documentation at any time.

Unless otherwise indicated, PortSIP registered trademarks are registered in the United States and other territories. All registered trademarks recognized.

For further information contact PortSIP Solutions, Inc.

Table of Contents

NOTICE 2
Overview, Q & A 4
IVR SDK Initialize and register methods/functions 5
IVR SDK audio codec methods/functions 11
IVR SDK DTMF methods/functions 13
IVR SDK calls methods/functions 14
IVR SDK play wave file to remote side method/function 17
IVR SDK play backgroud wave file to remote side method/function 18
IVR SDK Receive DTMF method/function 20
IVR SDK Caller/Callee infomation method/function 22
IVR SDK Audio recording functions 23
IVR SDK register events 25
IVR SDK calls events 26
IVR SDK Transfer events 32
IVR SDK Received DTMF events 34

Overview, Q & A

This document describes the methods and callback events supported by PortSIP IVR SDK libraries.

All samples of the PortSIP IVR SDK are available at
http://www.portsip.com/downloads.html

The Sample source code for Visual basic available at
http://www.portsip.com/downloads/ivr/VBIVRSample.zip

The Sample source code for Visual Studio C# available at
http://www.portsip.com/downloads/ivr/CSharpIVRSample.zip

Q:How to compile & test the sample project ?

A:

1: Download the Sample source code and evaluation SDK(http://www.portsip.com/downloads/IVRVBSample.zip)and extract it to a directory, for example: c:\IVRVBSample.

2: Rebuild the sample and test it.

Q:How to create a new project with the PortSIP IVR SDK?

A:

1: Download the Sample source code and evaluation SDK(http://www.portsip.com/downloads/IVRVBSample.zip)and extract it to a directory, for example: c:\IVRVBSample.

2: Run the Visual Studio .NET and create a new Windows Application project

3: Copy the PortIVR.dll IVREventEx.dll files to your project bin\release and bin\debug directories.
(Note: these DLL files MUST in the same folder with your EXE file.)

4: Copy the "PortSIP" folder to your project folder and add it into your Solution.

5： Right click the project, choose "Properties", click "Build" tab, check the "Allow unsafe code" checkbox.
(Note: the point 5 just for C# only.)

Please send email to us: support@portsip.com to get technical support.

All methods/functions of SIP Core SDK can be found in the PortIVR.cs file.

IVR SDK Initialize and register methods/functions

```
public Boolean createCallbackHandlers()
```

Create the callback handlers. This function must called before *initialize* function.

```
public void releaseCallbackHandlers()
```

Release the callback handlers. This function must called after *uninitialized* function.

```
public Boolean initialize(TRANSPORT_TYPE transportType,
                          PORTSIP_LOG_LEVEL appLogLevel,
                          String logFilePath,
                          Int32 maximumLines,
                          String agent,
                          String STUNSever,
                          Int32 STUNServerPort,
                          out Int32 errorCode)
```

To initialize the PortIVR SDK.

Parameters

transportType	-	Transport of SIP, it can be set as TRANSPORT_UDP, TRANSPORT_TLS, TRANSPORT_TCP.
appLogLevel	-	Use to set the application log level, the levels defined in the PortSIPTypes.cs file, if enable the log, then will generate `PORTSIP_APP_LOGFILE_xxxx` log file.
LogFilePath	-	The log file path, the path(folder) MUST is exists.
maximumLines	-	This parameter allows maximum 100 lines.
Agent	-	The User-Agent header to insert in messages.
STUNServer	-	Stun server, it's optional and can be pass empty string if you are not use STUN server.
STUNServerPort	-	Stun server port, it will be ignored if the STUNServer is empty.
errorCode	-	If the function succeeds, this parameter value is 0. If the function fails, this value is a specific error code can be found in PortSIP_Errors.cs file.

Return Values

If the function succeeds, the return value is true otherwise is false.

Remarks

If want to use the TLS transport, you have to install the root certificate into the "Trusted Root Certification Authorities area." Usually the certificate was issued by your ITSP.

```
public Int32 setUserInfo(String userName,
                         String displayName,
                         String authName,
                         String password,
                         String localIp,
                         Int32 localSipPort,
                         String userDomain,
                         String SIPServer,
                         Int32 SIPServerPort,
                         String outboundServer,
                         Int32 outboundServerPort)
```

Set the user information into SDK.

Parameters

username - Account(User name) of the SIP, usually provided by an IP-Telephony service provider.

displayName - *The display name of user name, you can set it as your like, likes: "James Kend". It's optional.*

authName - Authorization user name (usually equals the username).

Password - Password.

localIP - The local computer IP address to bind (for example: 192.168.1.108), it will be using for send and receive SIP message and RTP packet. If pass this IP as the IPv6 format then the SDK will using IPv6.

localSIPPort - The SIP message transport listener port(for example: 5060).

userDomain - User domain; this parameter is optional and can be pass NULL if you are not use domain.

SIPServer - SIP proxy server IP or domain(for example: xx.xxx.xx.x or sip.xxx.com).

SIPServer Port - Port of the SIP proxy server, (for example: 5060).

outboundServer - Outbound proxy server (for example: sip.domain.com), it's optional and can be pass NULL if you are not use outbound server.

outboundServerPort - Outbound proxy server port, it will be ignored if the outboundServer is empty.

Return Values

If the function succeeds, the return value is 0. If the function fails, the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void unInitialize()
```

Un-initialize the IVR Core SDK and release resources.

```
public Int32 getLocalIP(Int32 index, StringBuilder ip)
```

To get local computer IP address

Parameters

index - The IP address index, for example, the PC has two NICss, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.

IP - A pointer to the buffer(StringBuilder) that receives the IP, buffer size don't let it less than 16.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 getLocalIP6(Int32 index, StringBuilder ip)
```

To get local computer IP address in IPv6 format.

Parameters

index - The IP address index, for example, the PC has two NICss, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.

IP - A pointer to the buffer(StringBuilder) that receives the IP, buffer size don't let it less than 64.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 getNICNums()
```

Obtain the NIC(Network Interface Card) numbers of PC.

Return Value

If the function succeeds, the return value is NIC numbers, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void setLicenseKey(String key)
```

Set the license key into SDK.

Parameters

key - The license key, provide by PortSIP Solutions, Inc. (http://www.portsip.com)

```
public Int32 enableSessionTimer(Int32 timerSeconds, SESSION_REFRESH_MODE refreshMode)
```

This function allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests. The repeated INVITE requests, or re-INVITEs, are sent during an active call leg to allow user agents (UA) or proxies to determine the status of a SIP session. Without this keepalive mechanism, proxies that remember incoming and outgoing requests (stateful proxies) may continue to retain call state needlessly. If a

UA fails to send a BYE message at the end of a session or if the BYE message is lost because of network problems, a stateful proxy does not know that the session has ended. The re-INVITES ensure that active sessions stay active and completed sessions are terminated.

Parameters

timerSeconds - The value of the refresh interval in seconds. Minimum requires 90 seconds.

refreshMode- Allow set the session refresh by UAC or UAS.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void disableSessionTimer()
```

Disable the session timer.

```
public void setSrtpPolicy(SRTP_POLICY srtPolicy)
```

Set the SRTP policy..

Parameters

srtpPolicy - The SRTP policy is defined as:
If set the policy as **SRTP_POLICY_NONE**, the SDK can receive the encrypted call(SRTP) and unencrypted call both, but can't place outgoing encrypted call.
If set the policy as **SRTP_POLICY_FORCE**, the SDK just allows receive encrypted Call and place outgoing encrypted call only.
If set the policy as **SRTP_POLICY_PREFER**, the SDK allows receive encrypted and uncrypted call, and allows place outgoing encrypted call and unencrypted call

```
public Int32 setRtpPortRange(Int32 minimumRtpAudioPort,
                             Int32 maximumRtpAudioPort,
                             Int32 minimumRtpVideoPort,
                             Int32 maximumRtpVideoPort)
```

This function allows set the RTP ports range for audio and video streaming.

Parameters

minimumRtpAudioPort - The minimum RTP port for audio stream.

maximumRtpAudioPort - The maximum RTP port for audio stream.

minimumRtpVideoPort - The minimum RTP port for video stream.

maxmumRtpVideoPort - The maximum RTP port for video stream.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

The ports range must is an even number, don't let the audio RTP port range lapped over with video RTP port range, otherwise the function will failed.

```
public Int32 setRtcpPortRange(Int32 minimumRtcpAudioPort,
                              Int32 maximumRtcpAudioPort,
                              Int32 minimumRtcpVideoPort,
                              Int32 maximumRtcpVideoPort)
```

This function allows set the RTCP ports range for audio and video streaming.

Parameters

minimumRtcpAudioPort - The minimum RTCP port for audio stream.

maximumRtcpAudioPort - The maximum RTCP port for audio stream.

minimumRtcpVideoPort - The minimum RTCP port for video stream.

maxmumRtcpVideoPort - The maximum RTCP port for video stream.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

Don't getting audio RTCP port range repetitious with video RTCP port range, otherwise the function failed.

Don't getting RTCP ports range getting repetitious with RTP port range, otherwise the function failed.

The SDK split port range for each line(session) base on the "maximumLines" parameter which of "initialize" function. for example, if the port range is 40, the maximum lines are 5, the port range for each line is 40/5 = 8. If the port range for each line is less than 4, the function fails.

Examples:

The "MaximumLines" parameter for "initialize" function is set as 8:

A:

setRtpPortRange(10000, 11000, 12000, 13000); // OK

setRtcpPortRange(10500, 11500, 14000, 15000);// Wrong, the RTCP ports range is repetitious with RTP port range.

setRtcpPortRange(11001, 11100, 14000, 15000);// OK

B:

setRtpPortRange(20001, 20035, 20036, 20089); // OK

setRtcpPortRange(19000, 19019, 14000, 15000);// Wrong, the (port range)/(maximumLines) is less than 4

setRtcpPortRange(19000, 19032, 14000, 15000);// OK

C:

setRtpPortRange(20001, 20035, 20029, 20089); // Wrong, the port range is repetitious

setRtcpPortRange(19000, 19019, 19019, 15000);// Wrong, t the port range is repetitious

```
public Int32 registerServer(Int32 expires)
```

Register to SIP proxy server.

Parameters

expires - Registration refresh Interval.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void unRegisterServer()
```

To un-register from the SIP proxy server.

IVR SDK audio codec methods/functions

```
public void addAudioCodec(AUDIOCODEC_TYPE codecType)
```

Add an audio codec.

Parameters

codecType - Audio codec type, defined in the PortSIPTypes.cs file.

```
void setAudioCodecPayloadType(AUDIOCODEC_TYPE codecType, Int32 payloadType)
```

Set the RTP payload type for dynamic audio codec.

Parameters

codecType - The audio codec type.

payloadType - The new RTP packet payload type.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 setAudioCodecParameter(AUDIOCODEC_TYPE codecType, String parameter)
```

This function use to set the audio codec parameter.

Parameters

codecType - The codec type which you want to set the parameter to it.

parameter - The parameter in string format.

Remarks

Example:

setAudioCodecParameter(AUDIOCODEC_AMR, "mode-set=0; octet-align=1; robust-sorting=0");

```
public void clearAudioCodec()
```

Remove all added audio codecs.

```
public Boolean isAudioCodecEmpty()
```

Detect added audio codecs.

Return Values

If no audio codec set, the return value is true, otherwise is false.

```
public void discardAudio(Boolean discardIncomingAudio, Boolean discardOutgoingAudio)
```

Discard the sending and received audio packets.

Parameters

discardIncomingAudio - Set to true the all received audio packets will be discard.

discardOutgoing - Set to true the all sending audio packets will be discard.

```
public Int32 setAudioSamples(Int32 ptime, Int32 maxptime)
```

This function set the audio capture sample which will be appears in the SDP of INVITE and 200 OK message as "ptime" and "maxptime" attribute.

Parameters

ptime - It's should be a multiple of 10, and between 10 and 60(included 10 and 60).

maxPtime - For the "maxptime" attribute, should be a multiple of 10, and between 10 and 60(included 10 and 60). *Can't less than "ptime".*

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

IVR SDK DTMF methods/functions

```
public void setDtmfSamples(Int32 samples)
```

To set the samples of DTMF tone.

Parameters

samples - Samples of the DTMF tone, default is 160.

```
public Int32 sendDtmf(Int32 sessionId, Char code)
```

Send DTMF tone.

Parameters

sessionId - Session ID of the call.

code - DTMF Digit (1, 2, 3, 4, 5, ... 0, *, #)

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

For example send the # DTMF tone: ***sendDtmf(mSIPCoreLib, 1, '#');***

IVR SDK calls methods/functions

```
public Int32 call(String callTo, Boolean sendSdp, out Int32 errorCode)
```

Make the call(invite).

Parameters

callee - The callee username(number), it can be "sip:usernumber@sipserver.com" or "usernumber" only.

sendSdp - To enable/disable send SDP with invite. If set to true, then invite will include SDP, if set to false, make invite without SDP.

errorCode - If the function succeeds, this parameter value is 0. If the function fails, this value is a specific error code can be found in PortSIP_Errors.cs file.

Return Value

If the function succeeds, the return value is a session ID to the call, otherwise is `INVALID_SESSION_ID(-1)`.

```
public Int32 rejectCall(Int32 sessionId, int code, String reason)
```

Reject incoming call.

Parameters

sessionId - Session ID of the call.

code - Rejected call status code.

reason - Rejected call status text(rejected reason).

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 answerCall(Int32 sessionId, Int32 waitSeconds)
```

Answer the incoming call.

Parameters

sessionId - Session ID of the call.

waitSeconds - In the SIP, if the callee is answered the call – send a 200 OK message to caller, the caller should reply an ACK message, if the callee received this ACK then means this call is established, otherwise the callee has to waiting the ACK to established the call.
If set `waitSeconds` as 0, then this function is Non-blocking and does not waiting the ACK after sent 200 OK message, it will returns immediately; If set it more than 0, this function will blocking and returns once received the ACK. If does not received the ACK in waitSeconds, then function will return once timeout.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in

PortSIP_Errors.cs file.

```
public Int32 terminateCall(Int32 sessionId)
```

Terminate the conversation(Hang up the call).

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 hold(Int32 sessionId)
```

To place a call on call.

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 unHold(Int32 sessionId)
```

Take off hold.

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 refer(Int32 sessionId, String referTo, Int32 waitSeconds)
```

Refer the call to another one.

Parameters

sessionId - Session ID of the call.
referTo - Target of the refer, it can be "sip:number@sipserver.com" or "number" only
waitSecond - wait for refer success.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

Example: transfer the call to sip:testuser12@sip.portsip.com:8000

refer(sessionId, sip:testuser12@sip.portsip.com:8000,10);

Or transfer the call to sip:test1@176.11.90.202:

refer(sessionId, "sip:test1@176.11.90.202",10);

You can download the demo AVI at: http://www.portsip.com/downloads/video/blindtransfer.rar, please use the Windows Media Player to play the AVI file after extracted, it will shows how to do the transfer.

```
public Int32 attendedRefer(Int32 sessionId, Int32 replaceSessionId, String referTo)
```

To make an attended refer.

Parameters

sessionId - Session ID of the call.
replaceSessionId - Session ID of the replace call.
referTo - Target of the refer, it can be "sip:number@sipserver.com" or "number" only

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

Please view the sample project to got more details. Or download the demo AVI at: http://www.portsip.com/downloads/video/attendedtransfer.rar, please use the Windows Media Player to play the AVI file after extracted, it will shows how to do the attended transfer.

IVR SDK play wave file to remote side method/function

```
public void startPlayFile(Int32 sessionId, String waveFile)
```

To set a wave file to play to the remote party, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

waveFile - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes: c:\\sample.wav

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void waitPlayFileEvent(Int32 sessionId, Int32 maxWaitMilliseconds)
```

To set a wave file to play to the remote party, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

maxWaitMilliseconds - Maximum waiting time.

Return Value

<0 – the return value is a specific error code can be found in PortSIP_Errors.cs file.

0 – playfile has completed without interruption

1 – Recv a dtmf event the method

2 – A playstop method has interrupted the method

ECoreIVRWaitingTimeOut - waiting time out

```
public Int32 playFile(Int32 sessionId, String waveFile, String stopTones)
```

To set a wave file to play to the remote party, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

waveFile - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes: c:\\sample.wav

szStopTones - Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including

the star (*) and pound sign(#), or a plus sign (+) to indicate all tones..

Return Value

<0 – An error, the return value is a specific error code can be found in PortSIP_Errors.cs file.

0 – playfile has completed without interruption

1 – A Stoptone has interrupted the method

2 – A playstop method has interrupted the method

Remark

This is synchronous function, it will blocking until file play end.

```
public void stopPlayFile(Int32 sessionId)
```

Stop a current play file on a call.

Parameters

sessionId - Session ID of the call

```
public Int32 getPlayFileDetail(Int32 sessionId, StringBuilder fileName,
            out Int32 elapseTimeSec, out Int32 totalTimeSec)
```

Get current play file detail.

Parameters

sessionId - Session ID of the call.

szFileName - Return current Playfile name.

elapseTimeSec - Return the elapsed playing time.

totalTimeSec - Return the file total duration.

```
public Int32 seekPlayFile(Int32 sessionId, Int32 offsetSec, Int32 seekOrigin)
```

Moves the play file pointer to a specified location.

Parameters

sessionId - Session ID of the call.

offsetSec - Number of second to offset from origin.

seekOrigin - osition from where offset is added. It is specified by one of the following constants defined:

0 Beginning of file

1 Current position of the file pointer

2 End of file

IVR SDK play backgroud wave file to remote side method/function

```
public void startPlayFileAsBackground(Int32 sessionId, String waveFile)
```

Play a background wave file to remote, the remote side will hearing this wave file as background sound in the call when call is established.

Parameters

sessionId - Session ID of the call.

`fileName` - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes: c:\\sample.wav

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public void stopPlayFileAsBackground(Int32 sessionId)
```

Stop a current background play file.

Parameters

sessionId - Session ID of the call.

IVR SDK Receive DTMF method/function

```
public Int32 getDigits(Int32 sessionId, out Int32 digitsCount, StringBuilder digitsBuff)
```

Returns any DTMF captured in a buffer.

Parameters

sessionId - Session ID of the call.

digitsCount - Return the received DTMF tones count.

digitsBuff - Return the received DTMF tones in string.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

Remark

This call is Non-blocking. The buffer is not cleared until clearDigits method is called.

```
public Int32 waitForDigits(Int32 sessionId,
                           out Int32 digitsCount,
                           StringBuilder digitsBuff,
                           String stopTones,
                           Int32 maxDigitsCount,
                           Int32 interDigitDelay,
                           Int32 maximumDelay)
```

Wait For Digits until get enough digits or time out.

Parameters

sessionId - Session ID of the call.

digitsCount - Return the received DTMF tones count.

digitsBuff - Return the received DTMF tones in string.

maxDigitsCount - Max recvive digits number.

interDigitDelay - Inter digit delay, Maximum time between receive two digits(sec.).

maximumDelay - Maximum digit delay, Maximum wait digits time(sec.).

Return Value

<0 – An error, a specific error code can be found in PortSIP_Errors.cs file.

0 – A Stoptone has interrupted the method

1 – Max digits count has received

2 – DTMF receive interval over interDigitDelay(sec.)

3 - DTMF receive total time over maximumDelay(sec.)

Remark

This is synchronous function, it will blocking until returns.

The buffer is not cleared before clearDigits method is called.

```
public Int32 clearDigits(Int32 sessionId)
```

Delete all received DTMF tones in the buffer.

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

IVR SDK Caller/Callee infomation method/function

```
public Int32 getCaller(Int32 sessionId, StringBuilder caller)
```

Get the caller of call.

Parameters

sessionId - Session ID of the call.

outCaller - Return the caller of call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 getCallee(Int32 sessionId, StringBuilder callee)
```

Get the callee of call.

Parameters

sessionId - Session ID of the call.

outCallee - Return the callee of call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

IVR SDK Audio recording functions

```
public Int32 startChannelRecording(Int32 sessionId,
                                   String filePath,
                                   String fileName,
                                   Boolean appendTimestamp,
                                   AUDIO_RECORDING_FILEFORMAT fileFormat,
                                   RECORD_MODE recordMode)
```

Start record the voice.

Parameters

sessionId - Session ID of the call.

filePath - File path, for example: c:\\audio

fileName - File name, for example: audiofile (without extended name)

appendTimestamp - If set as true then the SDK will append the timestamp to filename automatically.

fileFormat - File format, if pass it as ***FILEFORMAT_WAVE***, then will be record the audio into a file as WAV format; If pass it as ***FILEFORMAT_OGG***, then will be as OGG format; If pass it as ***FILEFORMAT_MP3***, then will be as MP3 format.

recordMode - Record file mode,.

RECORD_RECV – Only record receive(speaker) data.

RECORD_SEND - Only record Send(microphone) data.

RECORD_BOTH – Record receive/send data

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 stopChannelRecording(Int32 sessionId)
```

Stop the audio call recording.

Parameters

sessionId - Session ID of the call.

bEnable - If the value is set to true, then record is enabled. If set to false, record is disabled

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

```
public Int32 getChannelRecordedInfo(Int32 sessionId,
                                    StringBuilder recordedFilename,
                                    out Int32 durationMS)
```

Get recording Filename and duration time.

Parameters

sessionId - Session ID of the call.

recordedFilename - Return recorded filename.

durationMS - Return record duration time.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.cs file.

IVR SDK register events

```
public Int32 onRegisterSuccess(Int32 callbackObject, Int32 statusCode, String statusText)
```

When register to SIP Server was successfully registers, this event will be fired.

Parameters

callbackObject - This is the value which you passed when you "new" the *PortIVR* class

statusCode - The reason.

statusText - The code.

```
public Int32 onRegisterFailure(Int32 callbackObject, Int32 statusCode, String statusText)
```

When register to SIP server was failed, this event will be fired.

Parameters

callbackObject - This is the value which you passed when you "new" the *PortIVR* class

statusCode - The failure reason.

statusText - The failure code.

IVR SDK calls events

```
public Int32 onInviteBeginingForward(Int32 callbackObject, String forwardingTo)
```

If the *enableCallForwarding* method is called, when the call is incoming, the SDK will forward it automatically and fire this event.

Parameters

forwardingTo - The call forwarding target SIP URI.

```
public Int32 onInviteClosed(Int32 callbackObject, Int32 sessionId)
```

If the remote party was closed the call, then this event fired.

Parameters

sessionId - The session ID of this call conversation

```
public Int32 onInviteAnswered(Int32 callbackObject,
                              Int32 sessionId,
                              Boolean hasVideo,
                              Int32 statusCode,
                              String statusText,
                              String audioCodecName,
                              String videoCodecName)
```

If the remote party was answered the call, then this event fired.

Parameters

sessionId - The session ID of this call conversation

hasVideo - If this is the video call, then this parameter is true, otherwise is false.

statusCode - The answer code.

statusText - The answer reason.

audioCodecName - The audio codec name which is in used.

videoCodecName - The video codec name which is in used.

```
public Int32 onInviteFailure(Int32 callbackObject,
                             Int32 sessionId,
                             Int32 statusCode,
                             String statusText)
```

If the outgoing call is failed, then this event fired.

Parameters

sessionId - The session ID of this call conversation

statusCode - The failure code.

statusText - The failure reason.

```
public Int32 onInviteIncoming(Int32 callbackObject,
                              Int32 sessionId,
                              String caller,
                              String callerDisplayName,
                              String callee,
                              String calleeDisplayName,
                              String audioCodecName,
                              String videoCodecName,
                              Boolean hasVideo)
```

When an incoming call is detected, this event will fire.

Parameters

sessionId - The session ID of this call conversation

caller - The caller of incoming call

callerDisplayName - The display name of caller

callee - The callee of incoming call

calleeDisplayName - The display name of callee

audioCodecName - The audio codec name which is in used.

videoCodecName - The video codec name which is in used.

hasVideo - If this call has the video, then this parameter is true, otherwise is false.

```
public Int32 onInviteRinging(Int32 callbackObject,
                             Int32 sessionId,
                             Boolean hasEarlyMedia,
                             Boolean hasVideo,
                             String audioCodecName,
                             String videoCodecName)
```

If the out going call was ringing, then this event fired.

Parameters

sessionId - The session ID of this call conversation

hasEarlyMedia - If this parameter value is true, then the call has the early media detected and you do not need to

play a ring tone on the local computer. If value is false, then early media detection is unavailable

and you must play a ringing tone on the local computer to alert for call ringing.

hasVideo - If this is a video call, then this parameter is true, otherwise is false.

audioCodecName - The audio codec name which is in used.

videoCodecName - The video codec name which is in used.

```
public Int32 onInviteTrying(Int32 callbackObject, Int32 sessionId, String caller, String callee)
```

If the outgoing call was processing, this event fired.

Parameters

sessionId - The session ID of this call conversation.

caller - The caller.

callee - The callee.

```
public Int32 onInviteUpdated(Int32 callbackObject,
                             Int32 sessionId,
                             Boolean hasVideo,
                             String audioCodecName,
                             String videoCodecName
                            )
```

This event will be fired when remote party updated this call.

Parameters

sessionId - The session ID of this call conversation

hasvideo - If this is a video call, then this parameter is true, otherwise is false.

audioCodecName - The audio codec name which is used by the call.

videoCodecName - The video codec name which is used by the call.

```
public Int32 onACTVTransferFailure(Int32 callbackObject,
                                   Int32 sessionId,
                                   Int32 statusCode,
                                   String statusText)
```

When the refer call is failed, this event will be fired. The ACTV means Active. For example: A established the call with B, now A transfer B to C, C rejected this refer call, A will received this event. B will received ***PASVTransferFailure*** event.

Parameters

sessionId - The session ID of this call conversation

statusCode - The failure code.

statusText - The failure reason.

```
public Int32 onACTVTransferSuccess(Int32 callbackObject, Int32 sessionId)
```

When the refer call is succeeds, this event will be fired. The ACTV means Active. For example: A established the call with B, now A transfer B to C, C accepted this refer call, A will received this event. B will received ***PASVTransferSuccess*** event.

Parameters

sessionId - The session ID of this call conversation

```
public Int32 onInviteUACConnected(Int32 callbackObject,
                                  Int32 sessionId,
                                  Int32 statusCode,
                                  String statusText)
```

When UAC sent the ACK for 200 OK, this event will be fired.

Parameters

sessionId - The session ID of this call conversation

statusCode - The status code.

statusText - The status reason.

```
public Int32 onInviteUASConnected(Int32 callbackObject,
                                  Int32 sessionId,
                                  Int32 statusCode,
                                  String statusText)
```

When UAS received the ACK for 200 OK, this event will be fired.

Parameters

sessionId - The session ID of this call conversation

statusCode - The status code.

statusText - The status reason.

```
public Int32 onRecvDtmfTone(Int32 callbackObject, Int32 sessionId, Int32 tone)
```

This event will be fired when received a DTMF tone from remote side.

Parameters

sessionId - The session ID of this call conversation

tone - The DTMF tone value the values are:

1 - key 1

2 - key 2

3 - key 3

4 - key 4

5 - key 5

6 - key 6
7 - key 7
8 - key 8
9 - key 9
0 - key 0
10 - key *
11 - key #

```
public Int32 onRemoteHold(Int32 callbackObject, Int32 sessionId)
```

If the remote party was place the call on hold, then this event fired.

Parameters

sessionId - The session ID of this call conversation

```
public Int32 onRemoteUnHold(Int32 callbackObject, Int32 sessionId)
```

If the remote party was un-hold the call, then this event fired.

Parameters

sessionId - The session ID of this call conversation

IVR SDK Transfer events

```
public Int32 onPASVTransferSuccess(Int32 callbackObject, Int32 sessionId, Boolean hasVideo)
```

When the refer call is succeeds, this event will be fired. The PASV means Passive. For example: A established the call with B, now A transfer B to C, C accepted this refer call, B will received this event. A will received ***ACTVTransferSuccess*** event.

Parameters

sessionId - The session ID of this call conversation

hasVideo - If this transfer call has video, then this parameter value is true, otherwise is false.

```
public Int32 onPASVTransferFailure(Int32 callbackObject,
                                   Int32 sessionId,
                                   Int32 statusCode,
                                   String statusText)
```

When the refer call is failed, this event will be fired. The PASV means Passive. For example: A established the call with B, now A transfer B to C, C rejected this refer call, B will received this event. A will received ***ACTVTransferFailure*** event.

Parameters

sessionId - The session ID of this call conversation

statusCode - The failure code.

statusText - The failure reason.

```
public Int32 onTransferRinging(Int32 callbackObject, Int32 sessionId, Boolean hasVideo)
```

When the refer call is ringing, this event fired.

Parameters

sessionId - The session ID of this call conversation

hasVideo - If the ringing(183) include the video, then is parameter is true, otherwise is false.

```
public Int32 onTransferTrying(Int32 callbackObject, Int32 sessionId, String referTo)
```

When the refer call is processing, this event fired.

Parameters

sessionId - The session ID of this call conversation.

referTo - The refer target.

IVR SDK Received DTMF events

```
public Int32 onRecvDtmfTone(Int32 callbackObject, Int32 sessionId, Int32 tone)
```

This event will be fired when received a DTMF tone from remote side.

Parameters

sessionId - The session ID of this call conversation

tone - The DTMF tone value the values are:

1 - key 1

2 - key 2

3 - key 3

4 - key 4

5 - key 5

6 - key 6

7 - key 7

8 - key 8

9 - key 9

0 - key 0

10 - key *

11 - key #